Early bird APC Injection DETECTION

1) First look for `CreateProcess` with `CREATE_SUSPENDED` flag

`Implies` → A process is "held open" for modification

---

2) Monitor `NtWriteProcessMemory` or `WriteProcessMemory`

`Implies` → If the Source Process is different from Target process we have caught an outside process writing data into new process. Ex → Notepad.exe writes into csvc.exe

---

3) Monitor the func `QueueUserAPC`

`Implies` → Specifically, we look for an APC being sent to a thread that is ~~exit~~ still into "Suspended/Alertable" state. This means attempt was made to influence the thread before it officially started working

---

4) ~~to~~ look for the function `ntdll!NtTestAlert` → `Trap` for shellcode

`How it works` → This func naturally checks if there is a task pending in the thread. Now the EDR performs API hooking. The first few bytes of `*NtTestAlert` are standard windows instructions. The EDR goes into memory of the process and overwrites those first few bytes with a JMP (Jump instruction). Now when the thread tries to run *, it hits `JMP` ~~which~~ and is immediately sent to EDR's own inspection code

`NtTestAlert`

So, EDR → "You are checking for notes... do you have ~~any~~ a request for that?"

Thread → "Yes, I have an APC"

EDR → "Where is that APC? Oh it's pointing to unbacked (Private Memory) [Shellcode

The EDR now scans the specific address

Proof of Injection

then Inspects the target address of the pending APC. If that address point to a memory region marked as MEM_PRIVATE (Unbacked memory) with PAGE_EXECUTE_READ permissions [That means the APC will be executed] it confirms that malicious shellcode is attempting to run before the process's official entry point